RECEIVED
CENTRAL FAX CENTER
JUN 11 2008

AMENDMENTS TO THE SPECIFICATION

On page 17, please replace paragraph [0074] with the following amended paragraph:

[0074] The forth program represented as low-level code 102 is finally tokenized into fcode, or boot code 106 (126). The process of tokenizing with respect to the forth program comprises compiling the forth program into fcode. This process is described in detail in the Open Firmware standard. The following is an example of translating one or more blocks or fragments of code of a second programming language, e.g., Java, into one or more corresponding blocks or fragments of boot code of a first programming language, e.g., Forth. The example is described with respect to Java user class files, but the translator may translate any type of block or fragment of code of the second programming language into the one or more blocks of boot code of the first programming language.

On page 17, please insert, after paragraph [0074], the following paragraphs [0074.1]-[0074.3], which are supported by pages 10–11 of incorporated U.S. Provisional Application Serial No. 60/479,657 (e.g., *See* paragraph [0001] of Applicant's specification).

[0074.1] As described in more detail on pp. 10-11 of U.S. Provisional Application Serial No. 60/479,657, filed June 18, 2003, in one example the J2F compiler takes Java class files supplied by the user and compiles them into a Forth program. This may be the middle step of a 3-step compilation process: (1) from Java to Java bytecode, (2) Java bytecode to Forth source code, and (3) Forth source code to Forth virtual machine code (fcode). That is, the class files that J2F takes as input can be those that would be produced by any standard Java compiler. The output of J2F is Forth source code that is tokenized by the tokenizer to produce Forth virtual machine code (fcode).

[0074.2] Forth is a low-level stack-based language that resembles the Java Virtual Machine in many respects. The main similarity is that it is stack-based, therefore much of the straight-line code produced by J2F is a fairly direct translation of corresponding Java bytecode. For example, the Microsoft Java compiler, given the Java fragment:

```
int x = 1;
int y = 2;
int z = x + y;
```

produces the Java bytecode

```
iconst_1
istore_1
iconst_2
istore_2
iload_1
iload_2
iadd
istore_3
```

[0074.3] The J2F compiler translates this to the following corresponding Forth fragment:

```
1                              \ iconst_1
depth r@ - $replace            \ istore_1
2                              \ iconst_2
depth r@ - 1 + $replace        \ istore_2
depth r@ - 1 + pick            \ iload_1
depth r@ - 2 + pick            \ iload_2
+                              \ iadd
depth r@ - 2 + $replace        \ istore_3
```